Hindawi Publishing Corporation
The Scientific World Journal
Volume 2013, Article ID 963874, 11 pages
http://dx.doi.org/10.1155/2013/963874

Research Article

Heuristic Search for Planning with Different Forced Goal-Ordering Constraints

Jiangfeng Luo, Weiming Zhang, Jing Cui, Cheng Zhu, Jincai Huang, and Zhong Liu

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Jiangfeng Luo; nudtluojiangfeng@gmail.com

Received 3 May 2013; Accepted 11 June 2013

Academic Editors: W.-J. Hwang, S.-S. Liaw, and S. H. Rubin

Planning with forced goal-ordering (FGO) constraints has been proposed many times over the years, but there are still major difficulties in realizing these FGOs in plan generation. In certain planning domains, all the FGOs exist in the initial state. No matter which approach is adopted to achieve a subgoal, all the subgoals should be achieved in a given sequence from the initial state. Otherwise, the planning may arrive at a deadlock. For some other planning domains, there is no FGO in the initial state. However, FGO may occur during the planning process if certain subgoal is achieved by an inappropriate approach. This paper contributes to illustrate that it is the excludable constraints among the goal achievement operations (GAO) of different subgoals that introduce the FGOs into the planning problem, and planning with FGO is still a challenge for the heuristic search based planners. Then, a novel multistep forward search algorithm is proposed which can solve the planning problem with different FGOs efficiently.

1. Introduction

A large majority of real-world problems have interfering subgoals. How to effectively plan for the interfering subgoals, especially when there are forced goal-ordering (FGO) constraints, has been a long term focus. As the Goal Agenda Manager (GAM) [1] used in the FF planner [2] and the ordered landmarks [3, 4] introduced in the LAMA planner [5], quite a number of approaches have been proposed but the performance results have scarcely improved. This is because, if any of the FGO constraints is violated, forward search may arrive at a deadlock, from which there is no way to reach the goal state. However, the proposed approaches such as GAM and landmark cannot detect all the deadlocks exactly and the undiscovered deadlocks make a planning difficult. In this case, this paper proposes an approach that can automatically put right the planning process when it leads the search to a deadlock and significantly improve the planning efficiency.

Many real-world problems as in military, industrial, aviation, and space domains involve FGO constraints. An example is a naval platform which has to counter many incoming missiles with different weapons [6]. Firing weapons at one missile may interfere with the interception of others. Thus it can cause the naval platform to suffer from high probability of leaking if the missiles are countered in an incorrect order. Another example is a scenario of a robot rescue [7]. Each robot has a special ability, such as survivor search/transport, cleaning barriers, or medical distribution. The rescue tasks should be finished coordinately in constrained orderings with respect to a given environment. Certain robots only care about their own subgoals and achieving them too early may result in the failure of an entire military operation. Additionally, FGO constraint can be observed in a NASA scenario as a digger robot is allowed to dig the ground on mars only after a photograph robot has taken a picture of the site [8]. Along with the complex domain dependent constraints, the FGO constraint is one of the main challenges that a planner needs to overcome for the above problems.

Next, we first give the problem statement. Some definitions and properties are proposed to explain why FGOs occur for a planning problem. Then, a novel forward search algorithm is proposed to solve the planning with FGOs. Based on the evaluation, it can be concluded that planning with FGOs is still a challenge for current automatical planners but our method can solve it efficiency.

2. Problem Statement

Before introducing forced goal ordering (FGO), we first give the description of a planning problem (O, I, G) as given in [1].

In a planning problem definition, O is a finite set of ground actions with the STRIPS style (in this paper). For any $o \in O$, there is $o = (pre, add, del)$, where pre, add, and del are finite sets of ground atoms. pre is the precondition set under which the action is applicable. add and del are the atoms added or deleted after the execution of the action. I and G are the finite sets of ground atoms and represent the initial and goal state of the problem. For any given state s and action $o \in O$, the result of applying o to s is

$$result(s, o) = \begin{cases} s \cup add(o) \setminus del(o), & \text{if } pre(o) \subseteq s, \\ s, & \text{otherwise.} \end{cases} \tag{1}$$

For the action sequence $\{o_1, o_2, \ldots, o_n\}$ ($o_i \in O$, $1 \leqslant i \leqslant n$), there is

$$\begin{aligned} &result(s, \{o_1, o_2, \ldots, o_n\}) \\ &\quad = result(result(s, \{o_1, o_2, \ldots, o_{n-1}\}), o_n). \end{aligned} \tag{2}$$

The planning problem is finding a sequence of actions π, such that $G \subseteq result(I, \pi)$. The set of action sequence is defined as Π^O. A state s is *reachable* from the initial state if and only if $\exists \pi \in \Pi^O$, s.t., $s \subseteq result(I, \pi)$. Similarly, an atom f is *achievable* from a certain state s if and only if $f \in s$ or $\exists \pi \in \Pi^O$, s.t., $f \in result(s, \pi)$. An action o is *applicable* in a reachable state s if and only if $pre(o) \subseteq s$.

Definition 1 (forced goal ordering (FGO) [1]). For the planning problem (O, I, G), let $g, g' \in G$ be the atomic goals. We say that there is a forced ordering between g and g', written as $g' \prec g$, if and only if, for any state $s(g, \neg g')$, there is no plan π satisfying $g' \in result(s(g, \neg g'), \pi)$. $s(g, \neg g')$ represents a reachable state, in which g has just been achieved, but g' is false.

In any given state, an atomic goal remaining false means that the goal is not achieved. Definition 1 illustrates that forward search arrives at deadlock $s(g, \neg g')$ when there is $g' \prec g$ and the atomic goal g is achieved before g'. In some of the literatures [2, 9], $s(g, \neg g')$ is called a dead-end state too. During the planning process, forward search which violates any of the FGOs may lead the planning to a deadlock, from which there is no way to the goal state.

With respecting to the FGO, there is goal ordering defined as reasonable goal ordering (RGO) written as $g' \prec_r g$. There is a reasonable ordering between g and g', if and only if, for any reachable state $s(g, \neg g')$, there is no longer a plan that can achieve g' from $s(g, \neg g')$ without deleting g, at least temporarily [1]. So, if there is $g' \prec_r g$ and g is achieved before g', in order to get a plan solution, the planning must first delete the achieved goal g, then to achieve g', and last to achieve g again. In this paper, as to focus the problem on FGO, it is supposed that there is no goal deletion during the planning process. Each goal cannot be deleted once it has been added.

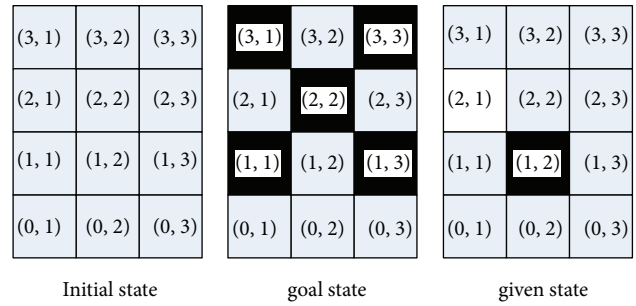

Figure 1: Floortile domain in IPC 2011.

The first planning domain for planning competition with FGOs is the Floortile proposed in the International Planning Competition (IPC) 2011 [10]. During the competition, no participating planner in the sequential satisficing track can solve it well. Figure 1 shows an example of the Floortile domain in the IPC 2011. In the initial state, the status of all floor tiles is *clear*. Floor tiles need to be painted black and white, while adjacent tiles should have different colors. Robots can only paint tiles that are in front (up) or behind (down). Moreover, once a tile is painted, a robot cannot stand on it. This particular configuration makes the domain very hard to solve because of the existence of FGOs. For example, suppose a robot first selects the tile(2,1) to paint in white. In further planning steps, the robot can only stand on a tile(2,1) to paint the front tile(3,1) in black. This process can be achieved if and only if the atom (robot-at tile(2,1)) is true. However, this atom is not true and cannot be added once the tile(2,1) has been painted. The reason is that the atom (robot-at tile(2,1)) is mutually exclusive with the atomic goal (painted tile(2,1) white), and (painted tile(2,1) white) cannot be deleted once it has been added. Therefore, painting tile (2,1) before (3,1) violates the FGO constraint, in consequence, causing the search to arrive at a deadlock.

In this example, there are many FGOs in the Floortile problem, and the robots should paint tiles obeying a correct sequence. In the case of Figure 1, the FGOs are

(*painted tile(3,1) black*) $\prec$ (*painted tile(2,1) white*) $\prec$ (*painted tile(1,1) black*);

(*painted tile(3,2) white*) $\prec$ (*painted tile(2,2) black*) $\prec$ (*painted tile(1,2) white*);

(*painted tile(3,3) black*) $\prec$ (*painted tile(2,3) white*) $\prec$ (*painted tile(1,3) black*).

In the above domain, all the FGOs exist in the initial state. No matter which approach is adopted to achieve an atomic goal, all the atomic goals should be achieved in a given sequence starting from the initial state. Otherwise, the planning may arrive at a deadlock. However, in some real-world planning problem, there is no FGO in the initial state. For any $g, g' \in G$, planning starting from the initial state to firstly achieve g or g' would not lead to a deadlock. However, the planning may arrive at a given state $s(\neg g_1, \neg g_2, \neg g_3)$ ($g_1, g_2, g_3 \in G$) without FGO among g_1, g_2, and g_3. Now,

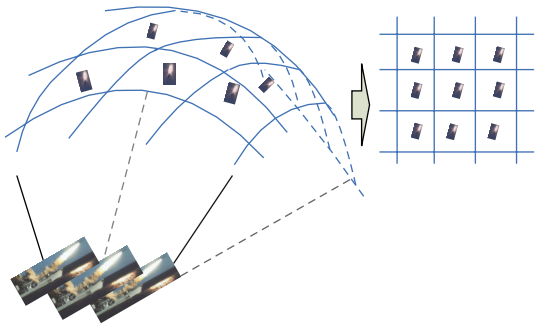

Figure 2: Air defense of a naval group.

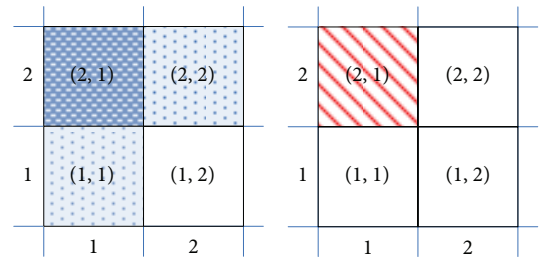

Figure 3: Air defense of naval group.

if certain plan as π_1 is selected to achieve g_1 while translating the search to state $s(g_1, \neg g_2, \neg g_3)$, then there is $g_2 \prec g_3$ or $g_3 \prec g_2$.

Figure 2 shows an example for the air defense of a naval group (ADoNG). A naval group has some Surface to Air Missile (SAMs) and chaffs to intercept the incoming antiship missiles. The arrived antiship missiles at the same time is supposed to locate at a spherical surface above the naval group, which can be transferred into a $m \times n$ rectangular plane as shown in Figure 2. In each rectangle, there is an antiship missile, while one ship of the naval group can fire a SAM or chaff to intercept it. However, once a chaff is employed to intercept an antiship missile in a given rectangle, the rectangles in the up, down, left, and right of the given rectangle should be interfered by the chaff cloud. If certain rectangle is interfered by the chaff cloud from one direction, the antiship missile in this interfered rectangle can only be intercepted by chaff, because chaff cloud can prevent the radar from guiding the SAM interception. Moreover, if certain rectangle is interfered by the chaff cloud from more than one direction, the antiship missile in this rectangle cannot be intercepted, because the radar of the naval group may lose the accurate position of the antiship missile. The goal state is to intercept all the incoming missiles by the given SAMs and chaffs.

For the planning of the air defense of a naval group, there is no FGO in the initial state. Taking the problem shown in Figure 3 as an example, there are 4 antiship missiles in a 2×2 rectangular plane with 3 chaffs and 1 SAM. Obviously, in the initial state, every antiship missile can be intercepted with the highest priority. However, if the antiship missile in rectangle $(2, 1)$ is firstly intercepted by a chaff, there are FGOs $(1, 2) \prec (1, 1)$ and $(1, 2) \prec (2, 2)$ in the successor state, and the antiship missile in (1, 2) must be intercepted by a SAM. However, if the antiship missile in rectangle $(2, 1)$ is firstly intercepted by a SAM, there is no FGO for the remaining antiship missiles in (1,1), (1,2), and (2,2).

3. Related Works

Heuristic search planning (HSP) has become a dominant domain independent paradigm over the last decade [5]. The HSP method, first proposed by Bonet and Geffner [11], performs a forward search from an initial state to a goal state in a search graph. This method employs some powerful heuristic estimators to guide the search in a fast forward manner towards the goal state, with the help of heuristics for choosing helpful actions to extend the search closer to the goal state. In the past, great success has been achieved in heuristic search planning systems, such as FF [2], FD [12], SGPlan [13], and LAMA [5]. Researchers have also considered various goal interactions in multiple-goal achievement and detection [1, 14]. All of the above planners have their own approaches to deal with the goal orderings.

Among the previous works, the most relevant methods to our approach are Goal Agenda Manager (GAM) and ordering landmarks. The concept of a Goal Agenda Manager (GAM) was proposed in [1] to detect the reasonable goal orderings. The GAM is widely used in many planning systems such as IPP and FF, which can improve the performances of IPP and FF dramatically. A GAM defines the order in which the subgoals are achieved. In the beginning of a search process, a GAM is employed to check all the ordering relationships of each atomic goal pair. Then, the search divides the goal set into many subsets so that the planner can achieve each of them in sequence.

For each atomic goal pair as $g, g' \in G$, GAM uses $F_{Dg}^{g} = \bigcap_{o \in O,\, g \in add(o)} del(o)$ and $O^* = O_g \setminus \{o \in O \mid pre(o) \cap F_{Dg}^{g} = \emptyset\}$, where $O_g = \{o \in O \mid g \notin del(o)\}$, to calculate whether there is an action sequence $p \in P^{O^*}$ satisfying $g' \in result(s(g, \neg g'), p)$. If there is not, there exists an ordering defined as $g' \prec_r g$. Otherwise, the GAM checks whether $g \prec_r g'$ exists. Therefore, there are $P_{|G|}^2$ atomic goal pairs that need to be checked.

A concept known as "landmark" is defined to extend the GAM on goal ordering at top-level atomic goals as well as certain states known as landmarks during a planning process [3, 4]. For a planning problem (O, I, G), an atom l is called a landmark if, for any $p = \{o_1, o_2, \ldots, o_n\} \in P^O$ and $G \subseteq result(I, p)$, there is $l \in result(I, \{o_1, \ldots, o_i\})$ $(1 \leqslant i \leqslant n)$. A planning system that uses landmarks obtains all the landmarks of the problem at the beginning of the planning process. The planner then orders them heuristically. It uses a backtracking method via a relaxed plan graph (RPG) [2] to find the candidate landmarks and their orders. For example, all the atomic goals are landmarks. For each atomic goal g, the atoms in $\bigcap_{\forall o \in O,\, g \in add(o)} pre(o)$ are treated as new landmarks, where there is relationship $f \prec_r g$ for each $f \in \bigcap_{\forall o \in O,\, g \in add(o)} pre(o)$. Then, f is treated as a new atomic

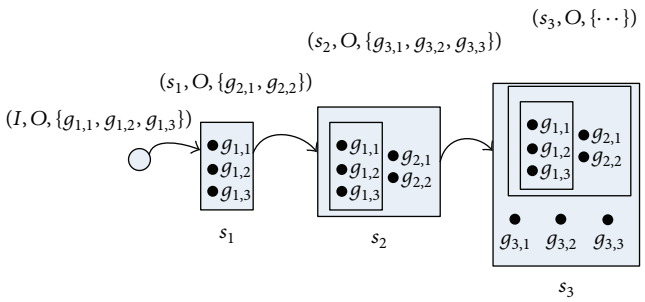

Figure 4: The incremental planning process.

goal. The landmark-generation algorithm repeats the above process from the top level of the RPG to the lowest level.

The third method to handle the goal ordering is the incremental planning process adopted by the planner SGPlan6 [13, 15]. As shown in Figure 4, the incremental planning process try to check part of the goal orderings in the initial state. Some of the checked atomic goals which can be achieved with high priority are firstly handled. Then, the incremental planning process try to find out more other goal orderings in current state and achieve part of atomic goals with high priority. The planning continues the above process until all the atomic goals are achieved.

4. Why FGOs Occur?

Definition 2 (goal achievement operation (GAO)). For planning problem (O, I, G), $o \in O$ is a GAO, if $\exists g \in G$, satisfying $g \in add(o)$. o is written as o_g.

Definition 3 (available GAO). o_g is available in a given state s, if and only if $pre(o_g) \subseteq s$, or $\exists \pi \in \Pi^O$, s.t. $pre(o_g) \subseteq result(s, \pi)$.

For an unachieved atomic goal g in s, an available GAO as o_g represents a plan, written as π_{o_g}, while $g \in result(s, \pi_{o_g})$. So, if $pre(o_g) \subseteq s$, there is $\pi_{o_g} = \{o_g\}$. Otherwise, if $pre(o_g) \subseteq result(s, \pi)$, there is $\pi_{o_g} = \{\pi, o_g\}$.

Definition 4 (available GAO sequence). Suppose that $G' \subseteq G$ while $s(\neg G')$ is reachable, and there is at least one available GAO as $o_i \in O$ $(1 \leqslant i \leqslant |G'|)$ for each $g_i \in G'$ in $s(\neg G')$ ($|G'|$ is the number of atomic goals contained in G'). These GAOs can be ranked as an available GAO sequence if and only if

(1) these GAOs are ranked in a correct sequence $\{o_{j_1}, o_{j_2}, \ldots, o_{j_{|G'|}}\}$, $(j_i = 1, 2, \ldots, |G'|, j_k \neq j_l$ if $k \neq l$, $1 \leqslant i, k, l \leqslant |G'|)$;

(2) there are number of $|G'|$ corresponding plans $\{\pi_{j_1}, \pi_{j_2}, \ldots, \pi_{j_{|G'|}}\}$ such that

$$
\begin{aligned}
pre\left(o_{j_1}\right) &\subseteq result\left(s\left(\neg G'\right), \pi_{j_1}\right),\\
g_{j_1} &\in result\left(s\left(\neg G'\right), \left\{\pi_{j_1}, o_{j_1}\right\}\right) = s_1,\\
pre\left(o_{j_2}\right) &\subseteq result\left(s_1, \pi_{j_2}\right),\\
\left\{g_{j_2}, g_{j_1}\right\} &\subseteq result\left(s_1, \left\{\pi_{j_2}, o_{j_2}\right\}\right) = s_2,\\
&\vdots\\
pre\left(o_{j_{|G'|}}\right) &\subseteq result\left(s_{|G'|-1}, \pi_{j_{|G'|}}\right),\\
G' &\subseteq result\left(s_{|G'|-1}, \left\{p_{j_{|G'|}}, o_{j_{|G'|}}\right\}\right).
\end{aligned}
\tag{3}
$$

Property 1. For the planning problem (O, I, G) with FGO constraints, the reachable state s is not a deadlock if and only if there is at least an available GAO sequence for the maximum unachieved atomic goal set in s.

Proof. Based on Definition 4, suppose that the maximum unachieved atomic goal set in s is G'. Since goal deletion is not considered in this paper, there is a plan

$$
\pi = \left\{\pi_{j_1}, o_{j_1}, \pi_{j_2}, o_{j_2}, \ldots, \pi_{j_{|G'|}}, o_{j_{|G'|}}\right\}, \tag{4}
$$

s.t., $G \subseteq result(s(\neg G'), \pi)$. Therefore, $s(\neg G')$ is not a deadlock. Additionally, if s is not a deadlock, there must be at least one available GAO sequence in s. □

Definition 5 (excludable constraint of GAO). For a reachable state $s(\neg g_1, \neg g_2)$, there are available GAO o_{g_1} for g_1 and o_{g_2} for g_2. There is an excludable constraint in $s(\neg g_1, \neg g_2)$, written as $o_{g_1} \nrightarrow o_{g_2}$, if and only if o_{g_2} is unavailable in state $result(s(\neg g_1, \neg g_2), \pi_{o_{g_1}})$.

Definition 6 (excludable GAO set). In a reachable state $s(\neg G')$ $(G' \subseteq G)$, the excludable GAO set for the available GAO o_g $(g \in G')$ is written as $\widehat{O}_{o_g}(s(\neg G'))$, if and only if, (1) for any $o_g \nrightarrow o_{g'}$ $(g' \in G', g' \neq g)$, there is $o_{g'} \in \widehat{O}_{o_g}(s(\neg G'))$; (2) for any $o_{g'} \in \widehat{O}_{o_g}(s(\neg G'))$, there is $o_g \nrightarrow o_{g'}$; (3) for any $o_{g'} \notin \widehat{O}_{o_g}(s(\neg G'))$, there is not $o_g \nrightarrow o_{g'}$.

Definition 7 (equivalent GAO). In a reachable state $s(\neg g)$ $(g \in G)$, o_g^1 and o_g^2 are two different available GAOs for g. o_g^1 and o_g^2 are equivalent in $s(\neg g)$, written as $o_g^1 \cong o_g^2$, if and only if $\widehat{O}_{o_g^1}(s(\neg g)) = \widehat{O}_{o_g^2}(s(\neg g))$.

Definition 8 (equivalent State). Reachable states s_1 and s_2 are equivalent, written as $s_1 \cong s_2$, if and only if, for any $\pi \in \Pi^o$, there is $G \subseteq result(s_1, \pi)$, there must be $G \subseteq result(s_2, \pi)$ and vice versa.

In a reachable state $s(\neg g)$, for an available GAO o_g with $pre(o_g) \nsubseteq s(\neg g)$, there may be two different action sequences π_1 and π_2 satisfying $pre(o_g) \subseteq result(s(\neg g), \pi_1)$ and $pre(o_g) \subseteq result(s(\neg g), \pi_2)$. However, $result(s(\neg g), \pi_1)$ might not be equivalent with $result(s(\neg g), \pi_2)$. In this case, the atomic goal g achieved by the same GAO o_g may lead the search to two different nonequivalent states. Planning with this feature may increase the search space dramatically when handling the planning with FGOs. Therefore, with respect to the above case, we can define two different GAOs as

$o_g^{\pi_1}$ and $o_g^{\pi_2}$ to replace o_g to ensure that, in a given state, achieving an atomic goal by the same GAO should lead the search to the equivalent state. Namely, for a given reachable state $s(\neg g)$ and an available GAO o_g, for any $\pi_1, \pi_2 \in \Pi^O$, if there are $pre(o_g) \subseteq result(s(\neg g), \pi_1)$ and $pre(o_g) \subseteq result(s(\neg g), \pi_2)$, there must be $result(s(\neg g), \{\pi_1, o_g\}) \cong result(s(\neg g), \{\pi_2, o_g\})$.

Property 2. For a reachable sate $s(\neg G')$ ($G' \subseteq G$, $g \in G'$) and two available GAOs o_g^1 and o_g^2, there are $s_1 = result(s(\neg G'), \pi_{o_g^1})$ and $s_2 = result(s(\neg G'), \pi_{o_g^2})$. If both s_1 and s_2 are not deadlock and there is $o_g^1 \cong o_g^2$, there must be $s_1 \cong s_2$.

Proof. Suppose that $O_{G'}$ is the set of all possible available GAO sequences contained in $s(\neg G')$, $O^1_{G'\setminus g}$ is the set of all possible available GAO sequences contained in s_1, and $O^2_{G'\setminus g}$ is the set of all possible available GAO sequences contained in s_2. As o_g^1 and o_g^2 have the same excludable GAO set, there must be $O_{G'} \setminus O^1_{G'\setminus g} = O_{G'} \setminus O^2_{G'\setminus g}$. It can be inferred that $O^1_{G'\setminus g} = O^2_{G'\setminus g}$. As s_1 and s_2 have the same set of possible available GAO sequences, it can be declared that, for any available GAO sequence contained in $s_1(s_2)$, this available GAO sequence must be available in $s_2(s_1)$. So there is $s_1 \cong s_2$. □

Property 3. For the planning problem (O, I, G), $s(\neg g)$ is a reachable state. o_g^1 and o_g^2 are two different available GAOs of g in $s(\neg g)$ while $o_g^1 \cong o_g^2$. Starting from $s(\neg g)$, if selecting o_g^1 to achieve g leads the planning to state s_1 and selecting o_g^2 leads the planning to state s_2, then, for any $\neg g_1, \neg g_2 \in s_i$ ($i = 1, 2$), if there is $g_1 \prec g_2$ in $s_1(s_2)$, there must be $g_1 \prec g_2$ in $s_2(s_1)$.

Proof. Based on the Property 2, it can be inferred that s_1 and s_2 have the same set of possible available GAO sequences. So if there is $g_1 \prec g_2$ in s_1, there is no available GAO sequence contained in s_1 which can achieve g_2 before g_1. As s_1 and s_2 have the same set of possible available GAO sequences, so starting from s_2, g_2 cannot be achieved before g_1 too. So there is $g_1 \prec g_2$ in s_2. Obviously, by the same way, it can be inferred that, if there is $g_1 \prec g_2$ in s_2, there must be $g_1 \prec g_2$ in s_1. □

Property 3 illustrates that, during the planning process, selecting the equivalent GAO to achieve an atomic goal introduces the same possible FGOs into the planning.

Definition 9 (independent goal set). For the planning problem (O, I, G), the goal set G can be divided into k independent goal sets, written as $\{G_1, G_2, \ldots, G_K\}$ ($1 \leqslant k$), while $G_1 \cup G_2 \cup, \ldots, G_k = G$ and $G_i \cap G_j = \emptyset$ (for all $1 \leqslant i, j \leqslant k$). G_i and G_j are called independent with each other, if and only if, for any $g_i \in G_i$ and $g_j \in G_j$, there is no excludable constraint between the GAO of g_i and g_j in each reachable state.

Based on the above discussion, it can be concluded that FGOs occur just because there are excludable constraints among the GAOs of different atomic goals. Take the instance shown in Figure 1 as a example. In the initial state, the available GAO for atomic goal *(painted tile(3,1) black)* is *(paint-up robot1 tile(3,1) tile(2,1))*. The available GAOs for atomic goal *(painted tile(2,1) white)* are *(paint-up robot1 tile(2,1) tile(1,1))* and *(paint-down robot1 tile(2,1) tile(3,1))*, as there are

(paint-up robot1 tile(2,1) tile(1,1)) $\nrightarrow$ *(paint-up robot1 tile(3,1) tile(2,1))*;

(paint-down robot1 tile(2,1) tile(3,1)) $\nrightarrow$ *(paint-up robot1 tile(3,1) tile(2,1))*;

So there is *(painted tile(3,1) black)* $\prec$ *(painted tile(2,1) white)* in the initial state.

Furthermore, for the planning which has no FGO in the initial state, the excludable constraints among the GAOs may introduce FGOs into the planning process. As the example shown in Figure 3, in the initial state, each antiship missile can be intercepted by a SAM or chaff. However, as there are

(Chaff-Intercept (2,1)) $\nrightarrow$ *(SAM-Intercept (1,1))*

(Chaff-Intercept (2,1)) $\nrightarrow$ *(SAM-Intercept (2,2))*

the FGOs *(intercept (1,2))* $\prec$ *(intercept (1,1))* and *(intercept (1,2))* $\prec$ *(intercept (2,2))* occur.

Obviously, for the planning problem (O, I, G), if for all $g_i, g_j \in G$ ($i \neq j$) while g_i is independent with g_j as Definition 9 described, no FGO may occur during the planning process.

Next, a forward search algorithm is proposed based on the above discussion to solve the planning problem with different FGOs.

5. A Novel Forward Planning Algorithm

During the planning process, selecting a GAO with the bigger excludable GAO set to achieve an atomic goal has the higher probability to introduce FGOs into the planning. Generally, for the same atomic goal, a search algorithm prefers to select the GAO with the smaller excludable GAO set first to achieve it. However, for some planning problem, keeping to select the GAO with smaller excludable GAO set first may cause certain operation resource excessively consumed. In this case, the later planning process can only select the GAO with the bigger excludable GAO set to achieve each atomic goal. Then, planning may lead to a deadlock as the FGOs introduced by the excludable GAOs. Therefore, with respect to the planning algorithm proposed in this paper, the atomic goal as g_1 with the fewest number of available GAO is firstly selected to be achieved by an available GAO with the biggest excludable GAO set. Then, calculate the number of available GAO for the remaining unachieved atomic goals. The atomic goal, whose number of available GAO is decreased after the achievement of g_1, is selected to be achieved with high priority. Continuing the above process, if planning arrives at a state which contains an unachieved atomic goal without available GAO, move the achievement sequence of this atomic goal ahead and find an available GAO with the smallest excludable GAO set to achieve it, while ensuring that all the prior achieved goals can

also be achieved by the prior selected GAOs (or the equivalent GAOs).

The excludable GAO set based forward search algorithm *Ex_MsFS* (multi-step forward search) for the planning with FGOs is displayed in Algorithm 1. In the initial state s, the planning selects an atomic goal with the fewest number of available GAO to be achieved first (step 04). With respect to the selected atomic goal as g, the available GAO as o_g with the biggest excludable GAO set is selected with the highest priority to achieve g (step 08). Then, in the successor state s' after g achieved (step 09), there exist the following two cases. In case one, there is an unachieved atomic goal as g', which has no available GAO in s'. In this case, s' is a deadlock. All possible available GAOs for g' in s' are excluded by the GAOs which have been adopted to achieve the earlier selected atomic goals contained in L. So the achievement sequence of g' should be moved ahead (step 12). The detailed moving algorithm is lately discussed in Algorithm 2. For case two (step 20), if all unachieved goals in s' have available GAO, the atomic goals which have fewer available GAOs in s' than that in s should be selected to be achieved with high priority starting from s' by the depth-first search strategy (steps 23, 07).

The algorithm *Move_ahead* used in step 12 of Algorithm 1 is displayed in Algorithm 2. For the step 11 of Algorithm 1, when there is an atomic goal as g having no available GAO in current state, it means that certain achieved goals stored in list L (step 09 of Algorithm 1) should not be achieved before g. So the achievement sequence of g should be tried to move ahead. Suppose there are *num* elements contained in L. Each element is written as (s', o_g, s), which means that selecting the available GAO o_g in current state s to achieve goal g transfers the state to the successor s'. The atomic goals contained in the list L are achieved by the sequence from the head to the end. The algorithm tries to set g as the ith ($1 \leqslant i < num$) goal to be achieved (step 02). The atomic goals stored in L from the location index 1 to $i - 1$ are still achieved by the previously selected GAOs (steps 04–06). Then, it is the turn to select an available GAO to achieve g. As to ensure that all the atomic goals stored in list L from the location index i to *num* can still be achieved by its previously selected GAOs or the equivalent GAOs of the previously selected GAOs, the algorithm chooses an available GAO with the smallest excludable GAO set to achieve g (step 09–17). Now, the algorithm starts to check whether the atomic goals stored in list L from the location index i to *num* can still be achieved by its previously selected GAOs or whose equivalent GAOs (steps 21–35). If it is, it means that, with respect to the atomic goals in L, g can be taken as the ith goal to be achieved. So the algorithm returns *true* (steps 31–33). Otherwise, move g ahead as the $(i-1)$th goal to be achieved. If achieving g before all the atomic goals in list L still cannot ensure that the atomic goals stored in list L can be achieved by their previously selected GAOs or whose equivalent GAOs, the *Move_ahead* algorithm returns *false* (step 36).

Based on Algorithms 1 and 2 and the Definition 9, it can be inferred that, for each do_while loop (steps 06–24) of the algorithm *Ex_MsFS*, the achieved goals contained in each list L defined in step 03 of Algorithm 1 come from the same independent goal set. During the depth-first search process, all the atomic goals which are related with each selected goal (in step 04 of Algorithm 1) are stored in each list Q. Obviously, the atomic goals in each list Q are from the same independent goal set. For the moving ahead process displayed in Algorithm 2, the requirement that all the achieved goals in each list L can still be achieved by their previously selected GAO or their equivalent GAO after the achievement sequence of certain atomic goal is moved ahead, is to ensure that with respect to each list L, the elements having contained in list Q would not be changed during the moving ahead process. The reason is that the equivalent GAO makes an atomic goal lose the same number of available GAOs. So the depth-first process could not be inferred by the moving ahead process.

Figure 5 gives an example of the air defense for a naval group to illustrate the search process of *Ex_MsFS*. In the initial state, there are 6 antiship missiles. There are 5 chaffs and a SAM that can be used to intercept all the antiship missiles while the antiship missile in rectangle (2,1) must be intercepted by a chaff. As there is only one available GAO ($Chaff\_Inter(2, 1)$) for the missile in (2,1), *Ex_MsFS* selects the missile in (2,1) to firstly intercept by a chaff and transfers the state to s_1 (step 04 of Algorithm 1). Element $(s_1, Chaff\_Inter(2, 1), I)$ is pushed into list L (steps 08-09 of Algorithm 1). When the missile in (2,1) is intercepted by a chaff, the rectangles (1,1) and (2,2) are interfered by the chaff cloud coming from rectangle (2,1). So the missiles in (1,1) and (2,2) cannot be intercepted by SAM. The number of available GAOs for missiles in (1,1) and (2,2) in s_1 is fewer than that of in I. Therefore, missiles in (1,1) and (2,2) are pushed back into list Q (steps 22-23 of Algorithm 1) and assigned the higher priority to be intercepted during in the depth-first process. Suppose that missile in (2,2) is firstly selected to be intercepted by a chaff in state s_1 (step 07). Planning leads to s_2 and element $(s_2, Chaff\_Inter(2, 2), s_1)$ is pushed back into list L (step 09 of Figure 4). At the same time, missiles in (1,2) and (2,3) are pushed back into list Q (step 22 of Algorithm 1).

Continuing the above process, element $(s_3, Chaff\_Inter(2, 3), s_2)$ is pushed back into list L and missiles in (1,3) is pushed back into list Q. Now, the planning arrives at state s_3 and list Q pops back the missile in (1,3) to intercept. As rectangle (1,3) is interfered by the chaff cloud from rectangle (2,3), the missile in (3,1) can only be intercepted by a chaff. In this case, rectangle (2,3) should be interfered by the chaff cloud from rectangles (2,2) and (1,3) as the state s_4 shows. So there is no available GAO for the missile in (1,3), and the interception sequence of the missile in (1,3) needs to be moved ahead (steps 11-12 of Algorithm 1).

Now, the *move_ahead* algorithm tries to intercept the missile in (1,2) before that of in (1,3). It means to intercept the missile in (1,2) using ($Chaff\_Inter(1, 2)$) starting from state s_3. This selection may lead the search to state s_5, in which missiles in (1,1) and (1,3) cannot be intercepted. So the intercept sequence of missile in (1,2) should be further moved ahead. The *move_ahead* algorithm tries to intercept it before the missile in (2,3), which leads the planning to state s_6. Obviously, the missile in (1,1) cannot be intercepted starting

Input: planning problem (O, I, G).
Output: plan π with $G \subseteq result(I, p)$
01 $s = I, G' = \emptyset, \pi = \emptyset$;
02 ***do***{
03 $Q = \emptyset, L = \emptyset$;
04 select $\neg g \in s$ with the fewest number of available GAO;
05 Q.push_back((s, g)), $tem_s \leftarrow s$;
06 ***do***{
07 $s \leftarrow$ Q.pop_back().s, $g \leftarrow$ Q.pop_back().g;
08 select an available GAO o_g with the biggest excludable GAO set;
09 $s' \leftarrow result(s, \pi_{o_g})$; L.push_back((s', o, s));
10 ***for each*** $\neg g' \in s'$
11 ***if*** g' does not have available GAO in s'
12 ***if*** Move_ahead(L, g', L')==true;
13 L.clear(), $L \leftarrow L'$, $s' \leftarrow L$.back().s';
14 ***else***
15 ***if*** G'.contain(g')== true
16 ***return*** FAILURE;
17 ***else***
18 Q.clear(), L.clear(), G'.insert(g');
19 Q.push_back((tem_s, g')), ***go to*** step 05;
20 ***for each*** $\neg g' \in s'$
21 ***if*** Available GAOs of g' in s' is fewer than that of in s
22 ***if*** Q.contain((s', g'))==false
23 Q.push_back((s', g'));
24 }***while***$(Q \neq \emptyset)$
25 ***for*** $i = 0 : L$.size()-1
26 π.push_back($L[i].o$);
27 $s \leftarrow s'$, L.clear();
28 }***while***$(G \not\subseteq s)$
29 ***return*** π.

Algorithm 1: *Ex_MsFS* algorithm.

from s_6. So the missile in (1,2) should be intercepted before the missile in (2,2). It means intercepting the missile in (1,2) starting from s_1. Now, the available GAO ($SAM\_Inter(1, 2)$) with the smaller excludable GAO set is selected. Furthermore, all the missiles in (2,2), (2,3), and (1,3) can still be intercepted by its previously selected GAOs. After the moving ahead process, planning arrives at state s_8, from which the goal state can be arrived after the missile in (1,1) is intercepted by the last one chaff.

6. Discussion of the Problem and Algorithm

This section proposes some properties about the planning with FGOs and the search algorithm *Ex_MsFS*.

Property 4. The complexity for solving the planning with FGOs, all of which exist in the initial state and are irrelevant with the approach for each atomic goal to be achieved, is P_n^n, where $n = |G|$.

Proof. As all the FGOs exist in the initial state and are irrelevant to the approach for each atomic goal to be achieved, the search can arrive at the goal state if and only if all the atomic goals are achieved by a correct sequence. It is a complete permutation problem. So the complexity is P_n^n. □

Property 5. The complexity for solving the planning problem, in which there is no FGO in the initial state but FGOs would occur during the planning process if and only if certain goal is achieved by an inappropriate approach, is up to $P_n^n \cdot K^n$, where $n = |G|$ and K is the average number of approach that can be adopted to achieve each atomic goal.

Proof. In this case, all the atomic goals should be achieved in a correct sequence, and each atomic goal should be achieved by an appropriate approach. However, as there is no FGO in the initial state, not all the atomic goals need to take part in the complete permutation. So the complexity is up to $P_n^n \cdot K^n$. □

For the Floortile problem, the solving complexity is P_n^n while the air defense planning problem for a naval group is up to $P_n^n \cdot K^n$.

Property 6. For a solvable problem with FGOs, the *Ex_MsFS* algorithm is sufficient to returning a plan solution if the *move_ahead* algorithm returns *true* (step 12 of Algorithm 1) for each time it is called.

Proof. If the *move_ahead* algorithm returns *true* (step 12 of Algorithm 1) for each time that it is called, it means the search process which violates certain FGO constraints has been put

Input: Achieved goal sequence list L. Goal g whose achievement sequence needs to be moved ahead.
Output: New goal achievement sequence list L'

```
01  bool success_moved=false, num ← L.size();
02  for i = num:1
03     L' ← ∅;
04     for (k = 1; k ⩽ i − 1; k + +)
05        state_op_pair ← L[k];
06        L'.push_back(state_op_pair);
07     s1 ← L[i].s;
08     bool have_op=false;
09      O_g(s1) = {O_g^1(s1), O_g^2(s1), ..., O_g^m(s1)} is the available GAO
        set of g in s1. For ∀o, o' ∈ O_g^n(s1), there is o ≅ o'(1 ⩽ n ⩽ m).
        The GAO in set O_g^{n1}(s1) has the smaller excludable GAO
        set than that of in O_g^{n2}(s1) if there is 1 ⩽ n1 < n2 ⩽ m;
10     for l =1:m
11        select an available GAO o'_g from O_g^l(s1);
12        s2 = result(s1, π_{o'_g})
13        if (∃¬g'' ∈ s2, g'' has no available GAO in s2)
14           continue;
15        else
16           have_op=true;
17           break;
18     if(have_op==true)
19        L'.push_back((s2, o'_g, s1));
20        bool is_break =false;
21        for j = i : num
22           o ← L[j].o;
23           if(o or its equivalent GAO o' is available in s2)
24              tem_o ← o or tem_o ← o';
25              s' = s2;
26              s2 = result(s', π_{tem_o});
27              L'.push_back((s2, tem_o, s'));
28           else
29              is_break=true;
30              break;
31        if(is_break==false)
32           success_moved =true;
33           return success_moved;
34        else
35           continue;
36  return success_moved;
```

Algorithm 2: *Move_ahead*(L, g, L').

right. So the search process can arrive at the goal state and return a plan solution. □

For some planning problem, the search may arrive at certain state, from which no matter which, atomic goal is firstly selected to achieve, the search should arrive at a deadlock. In this case, the *move_ahead* algorithm always returns *false* and the *Ex_MsFS* algorithm cannot return the plan solution. Therefore, further works need to be done to extend the *Ex_MsFS* algorithm to a more general case.

Property 7. Evaluated by the Relaxed Graph [2], *Ex_MsFS* algorithm is an enforce hill-climbing algorithm, which climbs multiple steps each time.

Proof. Suppose $H(s)$ is the distance, calculated by the Relaxed Plan Graph, between the reachable state s and the goal state G, where $H(G) = 0$. In the estimator of Relaxed Graph,

$$H(s) = h_{g_1}(s) + h_{g_2}(s) + \cdots + h_{g_n}(s) - \sum_{O_s} (|o| - 1), \tag{5}$$

where $g_i \in G\ (1 \leqslant i \leqslant n)$ is the unachieved atomic goal in s, and $h_{g_i}(s)$ is the number of actions in the relaxed graph to achieve the atomic goal g_i starting from s. Set O_s contains actions shared by different atomic goals during their achievements in the relaxed graph, where $|o|$ is the frequency of o that has been shared. For each atomic goal as g_i selected in step 04 of Algorithm 1, the *Ex_MsFS* algorithm selects an

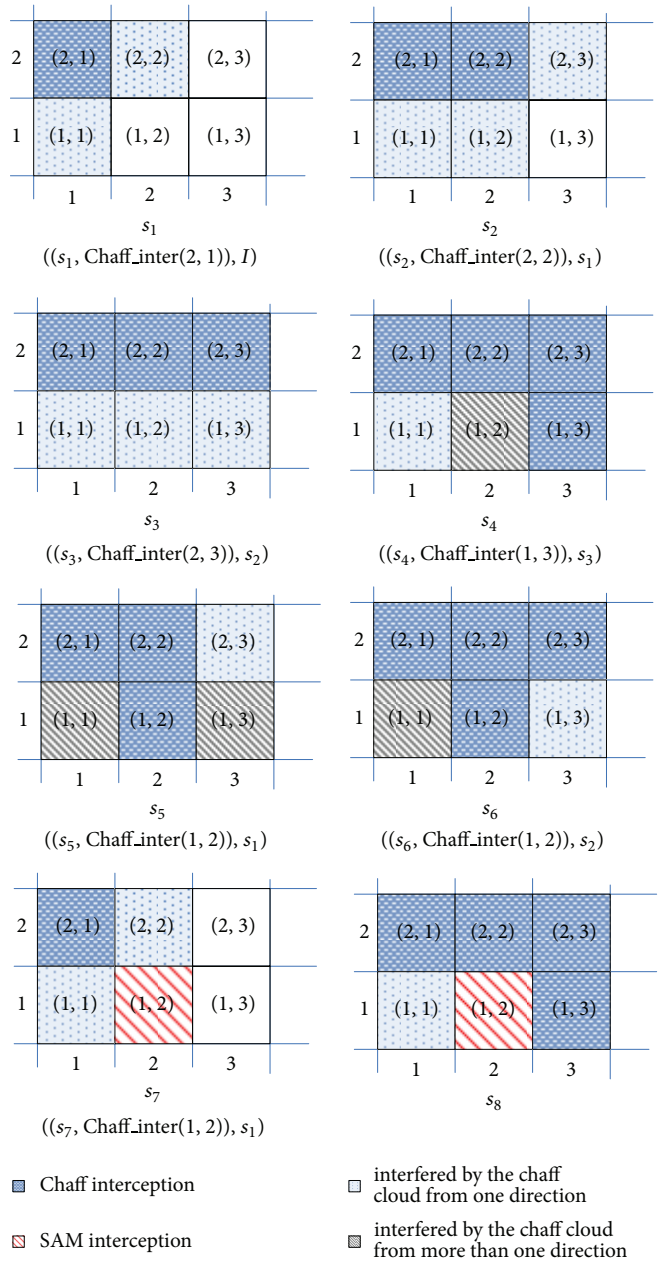

Figure 5: The example of air defense for a naval group.

available GAO and generates a plan to achieve g_i, which transfers the search from the current state s to the successor state s'. As action deletion is not considered in the relaxed graph, there are $h_{g_i}(s') = 0$ and

$$H(s) - H(s') = h_{g_i}(s) - \left(\sum_{O_s} (|o| - 1) - \sum_{O_{s'}} (|o| - 1) \right). \quad (6)$$

Suppose that g_i has k independent actions to other atomic goals in the Relaxed Graph starting from s. The value of $\sum_{O_s}(|o| - 1) - \sum_{O_{s'}}(|o| - 1)$ is $h_{g_i}(s) - k$. Therefore, for each selected atomic goal, there is $H(s) - H(s') = k$. The search direction of the *Ex_MsFS* algorithm is along the direction of the enforced hill climbing by k steps at a time toward a goal state. □

7. Evaluation

This paper evaluates the proposed algorithm *Ex_MsFS* by comparing it with the following planning systems:

(1) FF planning system [2], in which the GAM heuristic is used for the detection of goal ordering,

(2) SGPlan6 [13], which adopts the incremental planning process to handle the goal orderings,

(3) LAMA 2008 planning system [5], which adopts the hill-climbing strategy where the preferred actions of each step are selected by the FF and landmark heuristics. Moreover, the orderings of (disjunction) landmarks are calculated based on the domain transition graph and the causal graph.

FF, SGPLan6, and LAMA 2008 won the 1st prize of the satisficing planning track in IPC 2001, 2006, and 2008 respectively. In addition, the LAMA 2008 planning system consists of translating, processing, and searching modules. The translating and processing modules are used to construct some structured data, based on which, searching module is employed for forward search. In this paper, in order to compare the *Ex_MsFS* with the previous approaches, we use the *Ex_MsFS* algorithm to replace the search modules of the Lama 2008 planning system.

The competition domains are the Floortile proposed in IPC 2011 and the air defense of naval group problem introduced in Section 1. For an air defense of naval group problem, there are $m \times n$ antiship missiles while $|(m \times n)/3|$ SAMs and $m \times n - |(m \times n)/3|$ chaffs can be used. As this paper focuses on the efficiency when to solve a planning problem with FGOs, the quality of the planning solution is not considered. So we evaluate each planner by the scale of planning problems it can solve. The experiments are implemented in the Mac OS X with 2.4 GHz Intel Core 2 Duo and 2 GB 1067 MHz DDR3 memory. Each instance fails if the running time is more than 1,200 seconds.

Figure 6 shows the running time curves of the four different search approaches on the Floortile domain. There are 20 instances while LAMA 2008 can only solve 3 instances and FF can only solve 4 instances. However, SGPlan6 and the *Ex_MsFS* algorithm can solve all of the instances. Moreover, in this domain, SGPlan6 can solve each instance with the fewer time cost than *Ex_MsFS*. The reason is that the incremental planning process is very suitable in handling the FGOs of the Floortile domain. Taking the instance displayed in Figure 1 as an example, the incremental planning process first paints the upest tiles as (3,1), (3,2), and (3,3). Then, it paints the tiles (2,1), (2,2), and (2,3). At last, it paints the tiles (1,1), (1,2), and (1,3). This process satisfies the FGO constraints.

For the air defense of naval group problem, SGPlan6 can only solve only 1 out 0f the 17 instances while FF and LAMA 2008 can solve 10 instances. Moreover, as there are time constraints during the air defense process of the naval group in real world, if we require that each instance should be solved within 20 seconds, FF can solve only 7 instances and LAMA 2008 can only solve 6 instances. However, the *Ex_MsFS* approach can solve all of the 17 instances and return the solution plan for each instance within 10 seconds.

Based on Figure 7, it can be inferred that air defense of naval group problem is a new challenge for many current planners and *Ex_MsFS* can solve the planning problems with different FGOs efficiently.

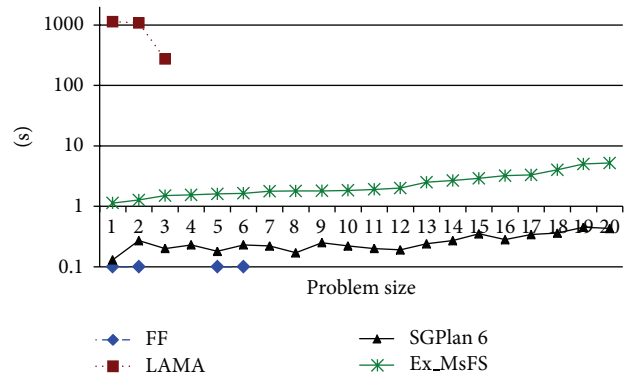

Figure 6: Planning results of the Floortile.

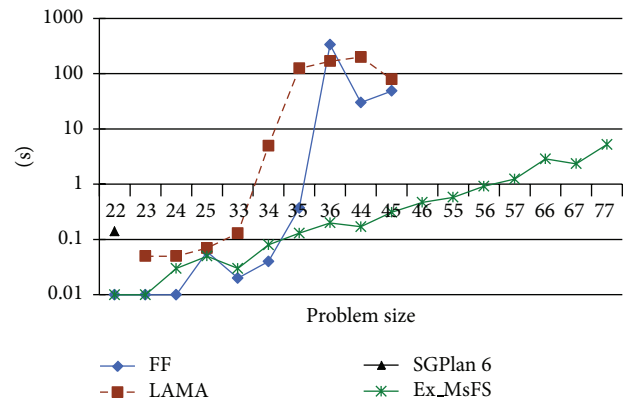

Figure 7: Planning results of air defense of naval group problem.

8. Conclusions and Future Works

This paper contributes to introduce a new planning domain with FGOs, for which all the current related planners do not perform well when solving it. Then, a new search algorithm is proposed which can solve the planning problem with different FGOs efficiently.

For future works, the *Ex_MsFS* should be extended to a more general case to solve more real-world problems with FGOs. Also, machine learning approach [16, 17] can be employed to obtain more informed about FGOs. The learned knowledge can improve the efficiency of the search process. Moreover, a human-computer method is considered for planning with FGO constraints. As some FGO constraints may easily be inferred by humans and other FGO constraints can be learned easily by computers, a human-computer method can be employed in a mixture of automatic and hand-crafted control rules [18]. In addition, if two different domains have similar subgoals interaction, the knowledge of deadlock checking leaned from one domain can be considered transferrable to other domains [19].

Acknowledgment

Thanks are due to the NSF of China which supported the research with Grants nos. 71001105, 91024006, and 61273322.

References

[1] J. Koehler and J. Hoffmann, "On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm," *Journal of Artificial Intelligence Research*, vol. 12, pp. 339–386, 2000.

[2] J. Hoffmann and B. Nebel, "The FF planning system: fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[3] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered landmarks in planning," *Journal of Artificial Intelligence Research*, vol. 22, pp. 215–278, 2004.

[4] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered landmarks in planning," *Journal of Artificial Intelligence Research*, vol. 22, pp. 215–278, 2004.

[5] S. Richter and M. Westphal, "The LAMA planner: guiding cost-based anytime planning with landmarks," *Journal of Artificial Intelligence Research*, vol. 39, pp. 127–177, 2010.

[6] A. Benaskeur, E. Bosse, and D. Blodgett, "Combat resource allocation planning in naval engagements," Tech. Rep., Valcartier, Canada, 2007.

[7] T. Morimoto, *How to Develop a RoboCupRescue Agent*, RoboCupRescue Technical Committee, 2000.

[8] A. Beynier and A.-I. Mouaddib, "Solving efficiently Decentralized MDPs with temporal and resource constraints," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 3, pp. 486–539, 2011.

[9] J. Hoffmann, "Where "ignoring delete lists" works: local search topology in planning benchmarks," *Journal of Artificial Intelligence Research*, vol. 24, pp. 685–758, 2005.

[10] T. De la Rosa, Floortile Domain, http://www.plg.inf.uc3m.es/, 2011.

[11] B. Bonet and H. Geffner, "HSP: heuristic search planner," in *AIPS Planning Competition Pittsburgh*, 1998.

[12] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[13] Y. Chen, B. W. Wah, and C.-W. Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan," *Journal of Artificial Intelligence Research*, vol. 26, pp. 323–369, 2006.

[14] D. H. Hu and Q. Yang, "CIGAR: concurrent and interleaving goal and activity recognition," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence and the 20th Innovative Applications of Artificial Intelligence Conference (AAAI '08)*, pp. 1363–1368, July 2008.

[15] C.-W. Hsu, Y. Chen, and B. W. Wah, "Subgoal ordering and granularity control for incremental planning," *International Journal on Artificial Intelligence Tools*, vol. 16, no. 4, pp. 707–723, 2007.

[16] Q. Yang, K. Wu, and Y. Jiang, "Learning action models from plan examples using weighted MAX-SAT," *Artificial Intelligence*, vol. 171, no. 2-3, pp. 107–143, 2007.

[17] J. Yin, X. Chai, and Q. Yang, "High-level goal recognition in a wireless LAN," in *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI '04)*, pp. 578–583, July 2004.

[18] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artificial Intelligence*, vol. 116, no. 1-2, pp. 123–191, 2000.

[19] D. H. Hu and Q. Yang, "Transfer learning for activity recognition via sensor mapping," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, Barcelona, Spain, 2011.